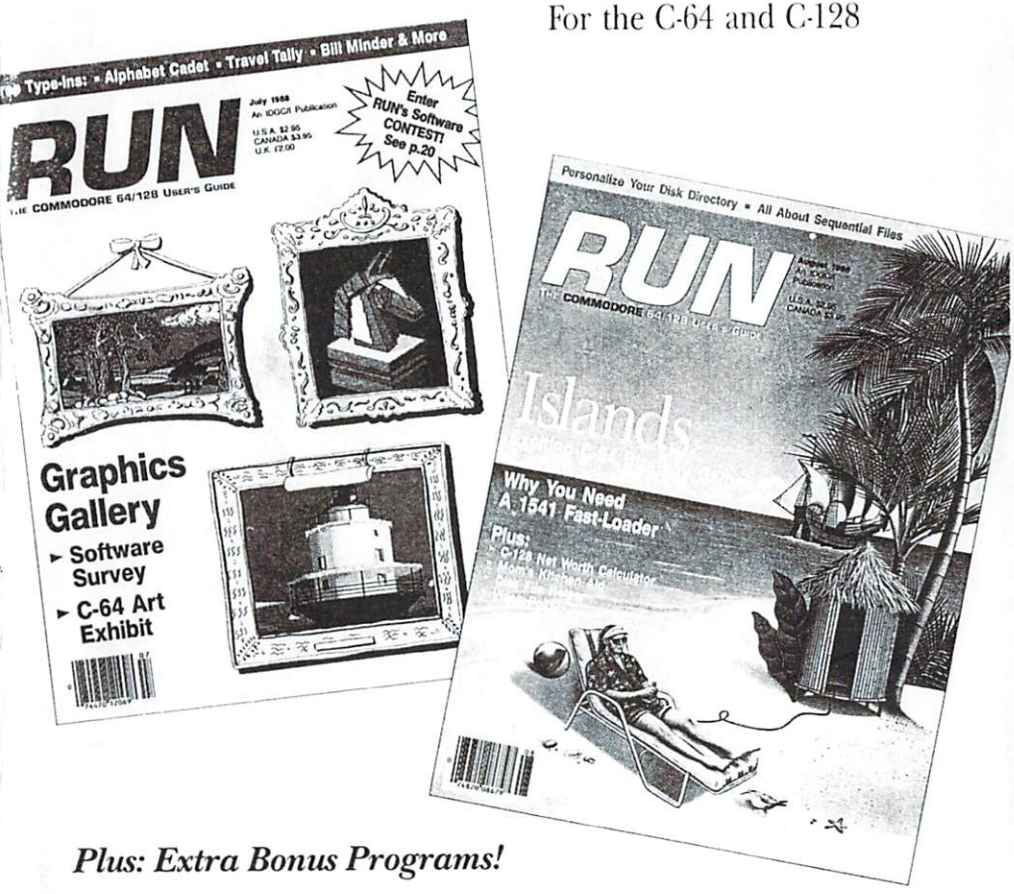
July/August 1988 Edition

RE RUN

RUN Programs on Disk

For the C-64 and C-128



Plus: Extra Bonus Programs!

Introduction

July-August '88 ReRUN

THIS JULY-AUGUST 1988 EDITION OF RERUN is our biggest and best in a long time, with no fewer than 18 programs. From the July issue, we have Alphabet Cadet, a spelling game in which you can compete against the computer, or two players can vie with one another. Random letters fall from the top of the screen, and you and your opponent race to shoot them down in order to spell a selected word.

The Sixteen-Color Print Machine, also from the July issue, is one of the most amazing programs I've yet seen for the C-64. Using four (blue, yellow, black and red) printer ribbons, you can create full-color printouts of Koala-format pictures on your Star Micronics or Epson printer.

Next on the list is Color Cornucopia, an 80-Column-mode C-128 program that enables your 1902 monitor to create an incredible number of colors. This is followed by another C-128 program, Bill Minder. Bill Minder won't reduce the number of bills you must pay, but at least it keeps accurate track of your expenses.

Travel Tally is yet another July program that should prove popular to many executives who travel on business and find filling out expense reports after the fact discouraging and frustrating. Travel Tally lets your C-64 help streamline the process of calculating travel expenses on a daily basis.

On the lighter side, there's Evel's Revenge, a C-64 game that calls upon your skills as a motorcycle barrel-jumper. July's Easy Application is Loan Analysis, a 64 or 128 loan calculator. Lastly, there's July's Mega-Magic, Disk Restorer, which allows you to restore data from accidentally formatted disks.

Heading the list of August programs is Islands, for the C-64, which is without a doubt one of the most dynamic two-player strategy games ever published. Two players using joysticks assume the roles of rulers of opposing Caribbean islands, and the goal is to build your island's population and your overall score beyond that of your opponent. Countless strategies are at your disposal to reach the goal of prosperity. You can build your island into a military fortress, a naval power or a well defended farming community.

Another worthy addition to the August issue is Net Worth Calculator, a C-128 program that allows you to to keep track of investments. Then, Video Poker brings Las Vegas and Atlantic City excitement to your C-64 by letting you gamble away your money. I've won over $800 in just a couple of attempts, only to lose it all in a few more turns.

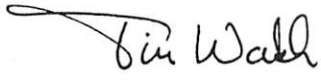
Disk File Helper is a C-64/C-128 menu-driven program that adds commands for retrieving deleted disk files and locking and unlocking files. You're sure to find it a worthy addition to your collection of disk utilities.

Another August disk-drive program is Sequential File Design, which shows you how to design sequential files. Yet a third disk program, The Remarkable Disk Directory, lets you add notes to disk directories so that you don't have to load a file into memory in order to determine its contents.

Wrapping up August, there's our Easy Applications program, Mom's Kitchen Aid, which is great for meal-planning and making grocery lists. Finally, August's Mega-Magic is Medium Resolution Graphics, a C-64 program that gives your computer a new programming mode.

Onward to our two bonus programs! The first is an 80-Column mode, C-128 utility called 1581 Directory. It makes for easy partitioning and accessing of those partitions on the 1581. The second is called Keys to Convenience, which lets you reconfigure the keys on your C-64. You can easily create a Dvorak keyboard or a numeric keypad using this program.

That wraps up this exciting edition of ReRUN. Stay tuned for the next edition's batch of programs—you won't be disappointed!



Technical Editor
RUN *magazine*

How To Load

Loading from Menu

To get started, C-64 users should type LOAD "MENU 64",8 and press the return key. When you get the Ready prompt, the menu is loaded and you should type RUN to see a list of the programs on your disk. C-128 users need only press the shift and run-stop keys. When all the programs are displayed on the screen, you can run the one you select by pressing a single key.

Loading from Keyboard

If you do not wish to use the menu program, follow these instructions.

C-64: To load a C-64 program written in Basic, type: LOAD "DISK FILENAME",8 and then press the return key. The drive will whir while the screen prints LOADING and then READY, with a flashing cursor beneath. Type RUN and press the return key. The program will then start running. To load a C-64 program written in machine language (ML), type: LOAD "DISK FILENAME",8,1

C-128: All C-64 programs can be run on the C-128 as long as your computer is in C-64 mode. All C-128 programs are clearly labeled on the directory page. Your C-128 *must* be in C-128 mode to run these programs. To load a C-128 mode program, press the F2 key, type the disk filename and then press the return key. When the program has loaded, type RUN.

Making Copies of ReRUN Disks

Many programs on your ReRUN disk have routines that require a separate disk onto which the program writes or saves subfiles. To use these programs, you must first make a copy of the original program onto another disk that has enough free space on it to hold these newly written subfiles.

It's simple to make a copy of a Basic program. Just load it into your computer as outlined above, and then save the program back onto a separate disk that has plenty of free space for extra files.

Copying an ML program is not so simple. You cannot simply load and save an ML program; you'll need to use a disk-backup utility program, such as the one on your Commodore Test Demo disk.

Directory

Page	Documentation	Disk Filename	File Type
		* MENU 128	BASIC
		MENU 64	BASIC
1	Alphbet Cadet	ALPHABET CADET	BASIC
		ALPHABET CADET 2	BASIC
		+ CADET.OBJ	ML
4	The Sixteen-Color Print machine	16-COLOR PRINTER	BASIC
7	Color Cornucopia	* COLOR CORNUCOPIA	BASIC
8	Bill Minder	* BILL MINDER 128	BASIC
9	Travel Tally	TRAVEL TALLY	BASIC
11	Evel's Revenge	EVEL'S REVENGE	BASIC
12	Loan Analysis	LOAN ANALYSIS	BASIC
13	Disk Restorer	RESTORER 1	BASIC
		RESTORER 2	BASIC
14	Islands	BOOT ISLANDS	BASIC
		+ ISLE ML	ML
		ISLANDS HEX LIST	BASIC
18	Net Worth Calculator	* NET WORTH CALC	BASIC
19	Video Poker	VIDEO POKER	BASIC
20	Disk File Helper	DISK FILE HELPER	BASIC
22	Sequential File Design	SEQ FILE DESIGN	BASIC
33	The Remarkable Disk Directory	REMARK-DIRECTORY	BASIC
34	Mom's Kitchen Aid	MOM'S KITCHEN	BASIC
35	Medium Resolution Graphics	MED RES GRAPHICS	BASIC
		MED RES DEMO	BASIC
36	£ 1581 Directory	* 1581 DIRECTORY	BASIC
39	£ Keys to Convenience	KEYBOARD EDITOR	BASIC

* – C-128 mode only
£ – Bonus program

Before you run a program, carefully read the documentation that pertains to it.

RUN it right: **C-64; one or two joysticks**

Alphabet Cadet

By John Ryan

ALPHABET CADET IS A VOCABULARY GAME that's so exciting the whole family will want to get involved. A one- or two-player game, it offers a high level of challenge and competition, with features including a scrolling playfield, sound effects, two levels of difficulty and full score-keeping, including high score.

The idea of the game is to shoot down letters that match those in a randomly selected word as an assortment of letters scrolls down from the top of the screen. While this may sound easy, competition for the letters is keen, because your opponent—either computer or human—is striving for the same letters you are. Level 2 offers an added twist: If a player shoots a letter that both players possess, the opponent will lose it from his score and must shoot for it again.

The first player to match the target word wins the round. Alphabet Cadet is programmed for five rounds of play, with 15 target words, but these defaults can be changed.

For two-player games, you'll need two joysticks. That of player 1 plugs into port 2 (the rear port), while player 2's joystick plugs into port 1 (yes, I know it's a little confusing). For one player, the single joystick goes into port 2 as well.

Playing Alphabet Cadet

Now you're ready to play. Load and run Alphabet Cadet. If all goes as it should, you'll see a colorful title screen.

After you specify the number of players and the difficulty level, the main Alphabet Cadet screen appears. The left half of the screen is the playfield, where random letters scroll down from top to bottom. Sprite "guns" are positioned at the bottom of the playfield, the red for player 1 and the blue for player 2. You can move these guns left and right within the playfield boundary once play starts.

The right half of the screen displays the scores. At the top part of this half is a block reserved for the target word, as well as color-coded red and blue scoring blocks for players 1 and 2. Round-

number and high-score information appears at the bottom of this half of the screen.

Press any key to begin. A target word appears in the block at the top of the scoring screen, then letters immediately begin scrolling downward toward the guns. When a letter appears that corresponds to one in the target word, position your gun directly beneath it and press the joystick button to fire. A successful hit moves the letter from the screen into your scoring block.

I stress "successful hit" for two reasons. First, just because you aim and fire at the letter doesn't guarantee you'll hit it. There are plenty of "duds" sprinkled throughout your arsenal, so be persistent. Second, your opponent is likely to be firing at the same letter. If he or she gets the letter first, you'll miss out on the points and, if you're playing at level 2, maybe a letter as well.

Level Play

In level 1 of Alphabet Cadet, geared toward younger players, the letters scroll down at a relatively slow pace, and your computer opponent targets only the letters that it needs. In level 2, the intensity of the game heats up considerably, with the computer opponent shooting like a maniac at all the letters it needs and every one you need as well!

In level 2 play, if both players possess a target-word letter, another hit on the letter removes it from the scoring block of the shooter's opponent, forcing him or her to fetch it again. Points are also awarded the shooter for removing an opponent's letter.

At level 2, you have essentially no time to study the board, especially when vying against the computer—which is one quick, intelligent and insensitive player. However, audio cues will help you keep abreast of what's occurring. When you or your opponent successfully shoots a letter, a low tone sounds, and when a letter is stricken from a player's scoring block, a high tone sounds. By listening to the tones, you'll soon know who's gaining and losing letters.

Winning the Game

The first player to match the target word wins the round and is awarded 100 points. In addition, each player earns points for the letters he or she possesses. Each letter has a value, with A worth the least and X worth the most. Bonus points are also added in, depending on the duration of the round—the longer it is, the more

bonus points awarded. These bonus points for lengthy games serve as an incentive at level 2, since it's extremely difficult to survive against the computer at that level.

At the end of five rounds, the points are tallied and the winner's name flashed repeatedly. Keep in mind that the player who wins the most rounds won't necessarily be the final victor, since the point total determines that.

Strategy

Young children, four to five years old, should be introduced to Alphabet Cadet with the two-player option, and just be allowed to shoot the letters as they appear on the screen, without trying to hurry or compete. Just matching the words without any antagonist is a joyful and rewarding experience for most youngsters. Single Player mode, with the computer providing competition, isn't recommended for kids in this age group, since the computer doesn't miss too often, and frustration may set in.

Once the child is confident with the mechanics of the game and can quickly identify all the letters in each word, he or she can progress to competing against another player in Two Player mode or the computer in One Player mode.

Older players can pursue various strategies. In a level 1 game, you can't strike letters from your opponent's scoring block, but that doesn't mean you should sit around waiting for your last letter or two while your opponent has only one more to go! Try to shoot down his or hers as well. While you won't get points for them, you'll force your opponent to wait for later appearances of the letters.

At level 2—well, you're on your own. With two players, the intense competition could make mortal enemies out of entire families. Against the computer, you'll get short-changed, slam-dunked, deep-sixed and generally get the stuffing beat out of you, but keep trying. Soon you'll be winning consistently, even if you're not a champion joystick athlete.

Customizing Alphabet Cadet

Adding more rounds per game, as well as more target words, is very easy. At the beginning of Alphabet Cadet, you'll notice the variable NW%, which represents the number of words, and NR% (number of rounds). You can set NW% to any number from 1 to 100, or even more, if you adjust the DIM statements. But never set NW% so that it is greater than the number of target words found

in the Data statements, or you'll get an Out of Data error.

NW% corresponds to the number of target words in the Data statements at the end of Alphabet Cadet. To add new target words, simply type the new words into the Data statements and change the variable NW% accordingly. Then resave the program. Target words must be no longer than eight characters, with no embedded spaces or non-alphabetic characters.

To change the number of rounds needed to win, just change NR% to whatever value you'd like.

Though Alphabet Cadet plays more like an arcade game than an educational one, you'll be surprised at how quickly youngsters will be identifying letters and learning new words with this program—that is, if they can get Mom and Pop off the computer long enough to play!

RUN it right: **C-64; Star- or Epson-compatible printer; Koala-format pictures; color ribbons; white paper**

The Sixteen-Color Print Machine

By Ted Davis

HOW OFTEN HAVE YOU WANTED to print out those beautiful, multicolor Koala-format pictures, but couldn't because a color printer is beyond your financial reach? Well, now you can do it by using this 16-Color Printer program with a Star Micronics- or Epson-compatible printer. You'll also need black, yellow, red and blue printer ribbons, some white paper and Koala-formatted pictures, either created by you or obtained from public domain sources, where many are available.

Sixteen-Color Printer reproduces the C-64 screen colors in much the same way that a variety of colors are printed in magazines and books: by layering black along with the primary colors—yellow, red and blue. An image is printed in each of these four colors separately

and in the proper density (percent of coverage) to produce the requisite combined effect. The program proceeds to translate the densities to paper by making a certain number of pins in the printhead matrix hit the ribbon. Table 1 lists the density of each printer color needed to produce each of the 16 screen colors.

Before you run the program, turn off the automatic linefeed on your printer, because the program produces its own. If you don't know how to do this, consult your printer manual for instructions.

When you run the program, a menu appears offering three options. F1 displays a directory of all the Koala pictures on a disk, each labeled with a letter from A to P. Pressing one of these keys loads the corresponding picture into memory. F3 displays on the screen the picture you've loaded; then any key returns you to the menu. F7 activates the print sequence for the picture currently in memory.

Table 1. Densities (in percent) of yellow, red, blue and black the program uses to reproduce each of the C-64's 16 colors.

C-64		Printer			
Number	Color	Yellow	Red	Blue	Black
0	Black	—	—	—	100
1	White	—	—	—	—
2	Red	—	100	—	—
3	Cyan	10	—	10	—
4	Purple	—	100	33	—
5	Green	100	—	33	—
6	Blue	—	—	100	—
7	Yellow	100	—	—	—
8	Orange	100	50	—	—
9	Brown	100	50	33	—
10	Lt. Red	10	50	—	—
11	Gray 1	—	—	—	66
12	Gray 2	—	—	—	33
13	Lt. Green	100	—	10	—
14	Lt. Blue	—	—	33	—
15	Gray 3	—	—	—	10

First Steps

The first phase of printing is the Picture Optimizer, which converts any secondary (defined but unused) colors to white. This will speed up the printing process. Press return to optimize or S to skip to the next step.

Now you need to insert your paper into the printer. Use at least two sheets (the thicker, the better) that are still attached. Although it's optional, in most cases you should clean the print head. You'll be using the yellow ribbon first, and ink on the pins will get it dirty. With paper inserted, remove your regular ribbon and press return to "print" five lines across the paper, removing ink in the process. Now you can place the yellow ribbon in the printer.

The final preparatory step is to align the paper in the printer, a procedure you must do precisely each time you print a different color, so the colors will line up properly. Position the perforation between two sheets just below the print head; then follow the screen prompts. Two short vertical lines will be printed near each other above the perforation. If they're far above, press L for a large adjustment. The paper will scroll a bit and each line will lengthen downwards. As the lines approach the perforation, press F for fine adjustment, then V for very fine adjustment. When the lines just touch the perforation, you're done. The lines are printed in slightly different places each time you align the paper for a different color, so you can see what you're doing.

With the paper properly positioned, press Y to print the yellow, which is the first color in line. When it's done, you'll be prompted to swap ribbons, then reinsert and realign the paper. (I don't recommend rolling the paper back.) Continue aligning and printing until all four colors are done. If you want to skip a color, press S instead of Y.

Remember that printing multicolor, hi-res graphics is slow, because the computer has to calculate and send the equivalent of 800 characters to the printer for each line of print. Pressing F1 during printing increases the speed slightly by turning off the screen display. You can increase speed a lot with a Basic compiler. I use Blitz, which works three to four times faster.

If you need public domain Koala pictures, refer to the list of public domain software sources on page 85 in last April's issue of *RUN* or send a $3 money order for shipping and handling to the Commodore Computer Club of Toledo, PO Box 8909, Toledo, OH 43623, attn. Ted Davis. Ask for PD Disk HE.

RUN it right: **C-128; Commodore 1902 monitor, or the equivalent**

Color Cornucopia

By Tim Walsh

IF YOU THOUGHT THE C-128 couldn't produce more than 16 colors, think again! It can make lots more—and all it needs is an 80-column RGB monitor, such as a Commodore 1902, that will work with the program (the 1902A and possibly other monitors will not), and the 80-Column Color Maker.

How many colors can it produce? Well, that's not easy to determine, but a conservative estimate would place the number at over three hundred distinct hues!

Experienced C-128 programmers are aware that you can scroll the contents of the 80-column screen horizontally. Eighty-Column Color Maker is a modification of the routine that does this scrolling, thus enabling the dazzling color cycling.

No programming tricks are involved. Color Maker draws bars on the screen of all the C-128's 16 colors and then scrolls them to the right. The colors wrap around to the left until the original screen display has scrolled completely off the screen to the right, whereupon the scrolling reverses and begins the trip back to the leftmost position.

As the display scrolls back and forth, the characters and background cycle through an incredible range of hues. In fact, so many colors appear that you'll find counting them futile.

When you do run it, pressing any key at any time will pause the action, so you can examine the varied colors in detail. Pressing any key will then resume the scrolling. To stop the program completely, just press the run-stop/restore combination.

Eighty-Column Color Maker is a fine utility to have at your disposal. You can use it as a routine to spice up text adventures, call attention to errors in application and utility programs, create Amiga-like title screens, and much more! Let your imagination run free—and when you come up with your own interesting ideas, let me know!

RUN it right: **C-128**

Bill Minder

By Jerome Reuter

DO YOU HAVE A CLEAR IDEA how much your expenses are going up—or down? If not, you're not in control of your finances. Bill Minder, my easy-to-use combination spreadsheet and database program, will help you get in control by organizing all your monthly bills and financial records and displaying them in various handy formats.

Written entirely in Basic 7.0, Bill Minder operates in the C-128's 80-Column mode. Each of the program's data files can store up to 12 months' worth of financial records, and you can keep as many files as you want by just changing data disks.

When you run the program, the first screen you'll see is blank except for a menu, listing the eight function keys, stretched across the top. The function keys are defined as follows:

F1. Loads the sequential data file, named DATA FILE.BM, from the disk.

F2. Saves an updated data file back to disk. Bill Minder also automatically saves a backup of the file. If you ever lose your work file, just rename the backup file DATA FILE.BM and load it with F1.

F3. Finds any dollar amount, checks the number or date; in addition, it searches for duplicates.

F4. Accesses the Enter mode, where you can type in new data, view the data that you've already entered and make changes.

F5. Displays a bar graph representing the monthly figures for any account (financial category) in your file. This option can reveal at a glance trends such as how much your electric bills went up last winter or what impact Christmas expenditures had on your credit card balance.

F6. Presents a tally sheet showing a total for each account for the months entered and a grand total for all your accounts.

F7. Produces a printout of any individual financial record, all the records for any month or any account for all the months entered.

F8. Creates a data file.

To exit Bill Minder, press any key other than a function key when you're at the main menu.

When you're setting up your "spreadbase" for the first time, select F8 and follow the prompts to input the names of your accounts and write your data file to disk. Then press F4 to access Enter mode.

In Enter mode, you'll see the accounts you named listed down the left side of the screen, column labels for amount, data, check number and memo across the top, and a place for monthly totals across the bottom. Each month occupies one page, and you can flip through the pages to add or correct records by pressing the N key. Unlike a spreadsheet, Bill Minder doesn't require you to move the cursor all around the screen to enter and correct data, and you never lose sight of your column headings and totals.

RUN it right: **C-64; printer optional**

Travel Tally

By Kenny Lawson

TRAVEL EXPENSE DIARY IS A SIMPLE C-64 database-type program for keeping a computerized record of expenses on business trips. All you need to do is enter the amounts you spend each day in the ten expense categories the program provides—hotels, dining, entertainment, transportation, and so on. If you want, you can change the categories in the Data statements to suit your needs.

For those who are interested in programming, the Data statements in lines 2210–2260 constitute a machine language routine that saves and restores text screens.

Making Entries

You should copy Travel Expense Diary to a work disk because the first time you run it, line 220 creates a dummy sequential disk file named Expense.Sq. Then the main menu, with six options, appears

on the screen. If you've used the program before and already have an Expense.Sq file, you press 1 at this point to load it and read in the latest totals. If you don't have a previous file, press 3, and the data-entry screen appears, showing the ten categories available. Use the cursor-up-and-down key to highlight to the category you want; then press the return key.

The return brings up a 3-D-effect window that prompts you to enter an amount. Only the return key, the number keys and the period (decimal point) are acceptable input here. If you press any other key, a warning sounds and a second 3-D window pops up, telling you to reenter your data.

The way to correct mistakes in your numeric input is to make an intentional invalid entry. For instance, say you press 6 when you mean 7; just press any letter key to access the reentry window. Pressing return with no input bypasses the entry of an amount into a category.

Once you've typed an amount, press return to add it to the previous total and restore the data-entry screen. Now you can either use the cursor and return keys to select another category or press M to go to the main menu. When you press M, each category into which you entered some amount will be automatically updated with a new total.

After returning to the menu, you might want to press 5 to see your totals on-screen or 4 to draw a colorful bar chart displaying the distribution pattern of your expenses. From option 4, you can also print out a hard copy of the bar chart, accompanied by a list of category totals and the grand total.

Program Notes

I've written several error traps into Travel Expense Diary. First, each disk input/output operation is checked, and any error is reported on-screen. Also, numeric data input is severely limited. In addition, if you've entered amounts into any categories, but haven't saved the file, the program reminds you of that fact before you exit. Then you can go ahead and exit, or go back to the menu to save the file first. In either case, when you press 6, the program asks if you're sure you want to quit.

As a final precaution, saves are done under a new filename, Bus.Seq.Del, so your old data is retained in Expense.Sq as insurance. If you accidentally save the file prematurely or make a mistake, all you have to do is exit the program, and, in Immediate mode, type in:

```
OPEN15,8,15,"S0:EXPENSE.SQ":CLOSE15
```

Then, after the light on the disk drive goes out, type in:

```
OPEN15,8,15,"R0:EXPENSE.SQ =BUS.SEQ.DEL":CLOSE15
```

In effect, this replaces the suspect file with the file that existed before the save. Finally, type in RUN to proceed normally.

Travel Expense Diary is written in Basic and occupies 7554 bytes of memory. You can customize it to suit other purposes by replacing the ten category names in the Data lines and changing the filenames where appropriate.

RUN it right: **C-64**

Evel's Revenge

By Curtis F. Kaylor

YOUR MOTORCYCLE SCREAMS down the track. Are you going too slow, too fast or just right for your launch into the sky and over the barrels? Well, no matter. If you miss, you won't break any bones—you're just playing Stunt Cycle.

When you run it, a motorcycle appears in the upper-left corner of the screen. If you have a joystick in port two, press the fire-button to accelerate; otherwise, press the space bar. To slow down, release the fire-button or space bar.

The speed at which you hit the launch ramp determines whether you'll clear the five barrels. If you're going too slowly, you'll fall short and crash; if you're going too fast, you'll overshoot and crash. Lose your concentration for any reason and you'll crash.

Eventually, you'll clear all five barrels and make a perfect landing. Then you can attempt six, seven and even more barrels, until you reach the ultimate jump—a 19-barrel grand finale. You get three motorcycles to start with and a fourth once you successfully clear the 19 barrels.

So, ladies and gentlemen, start your engines!

RUN it right: **C-64; C-128 (in 40- or 80-Column mode)**

Loan Analysis

By Lou Wallace

A HOME AND AN AUTOMOBILE are probably the largest purchases you'll ever make, and the chances are that you'll take out loans to finance them. When contemplating such a loan, it's important to know just what your monthly payments will be and how each installment will be divided between principal and interest. Loan Analysis, a C-64 and C-128 version of a standard loan analysis program, will produce such figures for any loan with a repayment period of at least one year.

When it's run, the program asks for the loan principal (amount borrowed), the interest rate and the term of the loan (how many years you'll be paying it back). It also asks for which year of the loan you'd like to see month-by-month figures and whether you want the output to go to the screen or the printer. When specifying the year, input 1 for the first, 2 for the second, and so on, not 1988 or 1989.

The output includes the amount of the monthly payments, a recap of the loan data you typed in and a table showing how much of each payment will go toward principal and how much toward interest during the year you requested, as well as the balance outstanding after each payment. Below the table you'll see the payment total for the year, along with how much of that total is principal and how much is interest.

Disk Restorer

By Kenny Lawson

THIS PAIR OF PROGRAMS CAN RETRIEVE FILES from disks you accidentally reformat with the OPEN15,8,15,"N0:<diskname> command. Save Directory (Listing 1) reads track 18 (the directory track) of a disk you want to protect and saves the data as a sequential file on a second disk. Later, if you accidentally reformat the protected disk, Restore Directory will write the sequential file back to track 18, thereby restoring the protected disk's files to use.

When you run Save Directory, it will prompt you to insert the disk you want to protect into the drive, press any key, then insert the disk that will hold the sequential file. The protected disk's name is used as the filename of the sequential file, and if there's a file of that name already on the sequential file disk, the program will ask if it's okay to delete it. If not, press N to exit the program; otherwise, press Y to go ahead and save the new file over the old one. Run Save Directory once for each disk you want to protect.

When you accidentally reformat a protected disk, load and run Restore Directory. Then insert the sequential file disk (if it's other than the disk containing these programs) and enter the name of the protected disk. The program reads in the sequential file with that name, then reminds you to place the protected disk in the drive. If you leave the file disk in the drive, you'll get two bad disks.

Note that these programs retrieve files only from disks reformatted with a short format command, which zeroes track 18 but leaves the files intact. If you reformat using a disk ID code, your files are actually erased and irretrievable.

Also bear in mind that changes to a disk after you save its directory won't be reflected in the sequential file. If a disk is important, resave its directory every time it's altered. Of course, if you don't resave the directory, Restore Directory can still recover some files.

In addition to restoring disk directories, these programs can be used to unscratch individual files on protected disks, as long as you do it right after the scratch operation. Just run Restore Directory.

RUN it right: **C-64; two joysticks**

Islands

By John Ryan

THE STORY GOES THAT, on the orders of King George III, two aristocratic brothers left London in the spring of 1799 and set sail for the Leeward Islands in the West Indies. The king was determined to turn the newly appropriated islands of Key Antigua and St. Christopher into profitable colonies, and the brothers, as governors, were given five years in which to do the job.

They arrived in the West Indies in early January of the following year and, each assuming control over one island, set about the task at hand. To all accounts, those early years were tough, with famine, hurricanes and rebel uprisings constantly threatening disaster.

Whether or not the governors met the king's expectations remains unclear, because records are fragmented. However, today the islands of Key Antigua and St. Christopher are thriving members of the British Commonwealth.

Welcome to Islands, an educational strategy game that puts you in the shoes of one of the governors, charged with building your island into a viable economic and military entity. This may be accomplished through treachery against the other island or honest hard work in nourishing your resources. Though Islands is meant to be played by two, you can also enjoy striving for a high score alone.

Starting To Build

Run the program "BOOT ISLANDS," and, after the title screen appears, press any key to start the game. Player 1 controls Key Antigua, the northern island, with a joystick plugged into port 2; player 2 governs St. Christopher, to the south, with the joystick in port 1.

Following the title screen, a map appears, and time starts ticking away. Now it's time to begin purchasing food, houses and factories by activating the appropriate icons under your Active label on the right side of the screen. You toggle these icons on and off by moving

the cursor against any screen border. Your score and information about your island's population and gold stores also appear on the right side of the screen, while a line at the bottom of the screen tells how many houses (H), crops (A) and defenses (D) you own at the beginning of each month.

To make a purchase, use the cursor to place the selected icon on the desired location and press the fire-button. If you have enough gold, the icon will be transferred to the island; if not, nothing will happen. Except for boats and troops, icons may be placed only on your own island. Boats, of course, go in the water, and you can place troops only on your opponent's island.

Carefully selected investments will build your simple island into a powerful nation. Your subjects must be fed and housed or productivity will go down and unrest may occur, and factories must be built to produce income.

The Icons

Crops, which are represented by green icons, cost five gold pieces (GPs) each, and a planting feeds 400 people. Financially, a crop is worth two gold pieces per month, plus ten GPs for each second of rain it receives. When you hear the rain falling, watch the gold grow!

At the end of each month, a portion of the crop is harvested, and then you can replant or cover the farmland with housing or factories. In deciding whether to plant or build, remember that the population of your island may be growing, and empty bellies cause unrest and hostility.

When island morale is low, revolutionary rebels (represented by shield and spear icons) rise up and often destroy factories and forts. Sometimes they're not visible, either; if you hear an explosion, and one of your factories or forts disappears, it's a good guess an unnamed rebel group was responsible. Once established on an island, rebels are only eliminated by attrition.

Houses cost 50 gold pieces and accommodate 500 people each. Morale and productivity will decline if your people get rained on, so don't let construction fall behind!

Factories cost 35 gold pieces each and generate a basic ten gold pieces per month. However, this amount is adjusted to reflect morale, hence productivity. If you overwork your laborers by establishing too many factories, their yield could plummet to zero. While factories produce income, they also increase an island's mortality rate through worker deaths.

Mines cost 100 gold pieces and generate 15–25 gold pieces per month, adjusted to morale and productivity. They, too, cause health-related deaths.

Hospitals, which cost 75 gold pieces, increase an island's productivity, add 10 percent to the monthly birth rate and add bonus points to your score.

Boats, at a cost of 25 gold pieces, serve as both income producers and warships. Each one generates five gold pieces per month in income, and a boat in Movement mode can contribute fishing income as well. When a boat that's fishing passes under a raincloud, its income increases, because the schools feed more in warm tropical rains. Listen to the coins drop into your coffer as the rain falls on the ocean!

Be careful, though; overfishing—resulting in no gold for either player—will result if both your boat and that of your opponent occupy the same area. Only boats in Movement mode may fish, and each one can feed 100 people.

Boats also add two points apiece to an island's defenses and may be used in naval warfare. You can sink enemy boats if they aren't in Movement mode and you have more defense points than your opponent. To sink a boat, put your warship into Movement mode,

Table 1. Quick Reference Chart.

Icon	Cost	Income	Affects. . .
Crops	5	2*	Welfare
Boats	25	5*	Welfare, Defense (2 points)
Factories	35	10	
Houses	50		Morale
Forts	60		Defense (10 points)**
Hospitals	75		Welfare, Morale
Mines	100	15–25	
Troops	150		Welfare, Morale, Defense***

* Additional income can accrue from rain or fishing.
** Forts protect all adjacent communities, industries and boats.
*** Destroy properties, thereby affecting these attributes.

P = Pause the game

move on top of the opposing boat and exit Movement mode. (See discussion of Forts, below, for an exception to this.)

To purchase a boat, select the appropriate icon, place the cursor over open water and press the fire-button. To shift a boat into Movement mode, activate the boat icon, place the cursor over the boat and press the fire-button. To exit Movement mode, press the fire-button a second time. The active icon may not be changed when a boat is in Movement mode.

Forts cost 60 gold pieces and add ten defense points to an island. These points are useful in naval warfare, as well as in combatting both rebels and invading enemy troops. Forts protect all adjacent crops, houses, hospitals, factories and gold mines from attacks by hostile forces. Likewise, boats anchored next to a fort cannot be sunk by your opponent's boats.

Invasion troops employed to attack your opponent cost 150 gold pieces each. To deploy your troops to destroy your opponent's industrial or housing projects, just place your cursor over the target and press the fire-button.

Note that defenses you add during any month aren't figured into your defense points until the beginning of the next month, and also that island defenses treat foreign troops and rebels in the same fashion.

WEATHER

The Caribbean enjoys warm and sunny weather for the most part, but constant tropical showers (gray clouds) do roam over the islands, accelerating the growth of crops. Hurricane season, from May until November, brings both hurricanes and lesser tropical storms (black clouds).

All weather systems start at the northwestern tip of Key Antigua, then move more or less southeast through the central part of the island and across St. Christopher. Because of this path, Key Antigua receives the most precipitation, while western sections of St. Christopher are arid and poorly suited to agriculture. This may seem an advantage to Key Antigua, but keep in mind that it also bears the brunt of hurricanes!

Tropical storms bring rain and winds, but only occasionally destroy industries. Hurricanes are unpredictable and devastating, usually destroying everything in their path. And, for each parcel of property destroyed, a corresponding portion of the population dies. Life in the tropics isn't always sunny!

Scoring

Islands' scoring is based on a complex socio/economic model that weighs not only current actions, but those taken many months past. You cannot switch from a military/industrial complex to an agrarian democracy and expect an instant increase in morale and productivity. You have five years in which to build up your island, at the end of which the player with the highest score wins. The maximum you can earn per month is 260 points, plus hospital bonuses.

Does ruling an island sound easy? Well, just remember that banana republics fall as easily as do crops during a hurricane!

RUN it right: **C-128 (80-Column mode)**

Net Worth Calculator

By Barbara Schulak

MY HUSBAND AND I have had to fill out personal financial statements for our bank several times in the past few years. Each time we've had to start from scratch, so when it came time again to update the form, I decided to write Net Worth Calculator for the C-128. Now we have a permanent record, and we can easily update it.

After you run the program, the computer will draw the page you'll be working on. Across the top of the screen is the menu bar. It contains the Load and Save options, Print, Assets, Liabilities, Calculate, Directory, Help (whose screen appears below the work page) and Exit. Under the bar are the Assets and Liabilities columns. Some of the assets listed are cash, stocks, real estate, vehicles and household goods. Liabilities include outstanding credit card balances, mortgages and notes payable to others. The space at the bottom of the screen is used for inputting data.

Figuring Your Net Worth

To begin, you'll want to create a new file; so, with the left-right cursor key, highlight the Assets option in the menu bar and press return. The first item in the Assets column will be highlighted. Use the up-down cursor to select an item for which you have data. Press

the plus-sign key, then type the item's value, which will appear in the cell next to the item description when you press return. Use the minus-sign and return keys to change an entry. Later, when you need to zero, or wipe out, a figure, use the Z and return keys. Entries should be in whole dollar amounts, and you needn't use commas in numbers; they are automatically inserted, if applicable, when you enter an amount.

Use the Help option any time you need to see the list of key controls needed to use the program, and press the no-scroll key in the top row of your keyboard to freeze the Help screen if it doesn't remain long enough. Press any key to deactivate the no-scroll.

When you've entered all of your assets, press return to go back to the menu bar and select Liability. Repeat the same procedure to enter or change figures in the Liability column. When you're done, return to the menu bar.

Now select Calc. It will automatically figure your total assets, liabilities and net worth.

Have a formatted work disk handy and use the Save option to save your work page. If you need a hard copy of your work page, choose the Print option and follow the prompts.

When you reload Net Worth Calculator, press return on the Load option and enter the filename of the work page you wish to use. If you don't remember the filename, use the View option to get a listing of the disk directory. Before leaving the program with the Exit option, be sure to save your file if you have added or made any changes to your data. You may also cancel any operation such as Load, Save or Print by entering return at the first prompt.

One more thing: You can enter values up to 999,999,999, although that's seldom useful. But who knows? Maybe you'll win a lottery!

RUN it right: **C-64; joystick**

Video Poker

By Tony Brantner

VIDEO POKER IS A ONE-PLAYER GAME for the C-64 that lets you try your luck at the draw without risking loss of your shirt. Written

in Basic and controlled with a joystick plugged into port 2, it takes about 30 seconds to initialize.

You start play with a balance of $100, then, at the beginning of each hand, place a bet of up to $50, but no more than your balance. Push the joystick forward to raise your bet and pull back to lower it.

Once you've settled on an amount, press the fire-button, and the cards at the top of the screen will turn over. To choose to hold any card, move the flashing cursor under it and press the fire-button. A "Hold" message will appear to indicate your choice. If you change your mind, press the fire-button again to clear the message.

After making your choices, pull the joystick back so that the cursor flashes on the "Draw" box and press the fire-button. The original cards you "discarded" will be replaced with new ones from the deck. After you "draw," the computer will evaluate your hand for the best combination and add any winnings to your balance. Payoff odds are shown at the bottom of the screen.

If you go broke or want to quit, place a bet of $0. The program then asks you if you want to play again. Push the joystick forward for yes, or pull it back for no, and press the fire-button.

RUN it right: **C-64 or C-128 in 40- or 80-Column mode; 1541 or 1571 disk drive**

Disk File Helper

By Bob Kodadek

DISK FILE HELPER IS A UTILITY that reprograms the 1541/1571 disk drive's operating system (DOS) to let you lock, unlock and unscratch files and change filetypes. It works with program (PRG), sequential (SEQ) and user (USR) files, and is much faster than using the usual track and sector editor, which spends time transferring blocks of data back and forth between the drive and computer.

Filetype Primer

Actually, PRG, SEQ and USR files are all sequential, differing only in manner of access and normal use. PRG files load directly into

memory, so are most often used to store Basic and machine language programs. SEQ files, which must be read from beginning to end, are used for data and text files, such as those generated by databases and word processors. USR files are similar to SEQ files, but their contents are ordinarily arranged in special formats for unusual applications.

The value stored in the filetype byte, which is the first byte in the directory entry, specifies the type (0–4) for all properly closed files. Table 1 lists these values in binary, hex and decimal for various types of files. Note that direct-access-type random files don't appear in the directory.

Using the Program

Whenever you need the assistance of Disk File Helper, just load and run it. The program's menu is shown in Table 2.

Table 1. Contents of the filetype byte for various kinds of files.

Type	Binary	Hex	Decimal
0 DEL (deleted)	10000000	$80	128
1 SEQ (sequential)	10000001	$81	129
2 PRG (program)	10000010	$82	130
3 USR (user)	10000011	$83	131
4 REL (relative)	10000100	$84	132

Table 2. Disk File Helper menu.

1. View the directory
2. Change the filetype
3. Unscratch a file
4. Unlock a file
5. Lock a file
6. Quit

Option 1 lists the disk directory to the screen. To freeze and unfreeze the scrolling, press and release the space bar.

Option 2 prompts you to enter the name of the file whose type you want to change, then displays a menu offering a choice of SEQ, PRG or USR. Being able to change the filetype is especially handy for loading SEQ and USR files into a monitor program for examination, since these files can't be loaded with the Load command.

Option 3 lets you unscratch, or restore, files you've inadvertently deleted from your disk directory, as long as you do it before another write or save operation to the disk. Just enter the name of the file, followed by the filetype you want it to have. It's possible to restore files, because scratching a file doesn't actually erase the data; it merely frees up the blocks reserved for that file in the block allocation map (BAM). Once scratched, and prior to being restored, those blocks are in danger of being overwritten and the file data destroyed. Unscratching with Disk File Helper changes the filetype byte back to its original status and then validates the disk. Incidentally, DEL (deleted) files don't appear in the disk directory, because DOS doesn't set bit 7 of their filetype byte.

To unlock or lock a file, choose menu option 4 or 5, respectively, then enter the filename at the prompt. Locking a file offers valuable protection from alteration and erasure, so it's surprising Commodore left it out.

Commodore 128 owners should note that a bug in the Burst Load routine in early 1571 ROMs prevents locked PRG files from being loaded. This problem, along with many others, has been corrected in the latest ROM release. The part number of the updated chip is 310654-04.

RUN it right: **C-64 and C-128; 1541/1571 or compatible disk drive**

Sequential File Design

By Steven Rogers

THERE ARE FOUR TYPES of disk files that can be saved on a 1541/1571 or other Commodore-compatible disk drive: program files,

sequential files, relative files and random files. Sequential and relative files are the types most often used for data storage, although program files are frequently used. Usually, some knowledge of assembly language programming is needed to write programs that store data in program files or random files. For that reason, these files are not covered in this article; we'll only examine sequential and relative files.

TERMINOLOGY

First, you must understand some terminology common to both sequential and relative files.

A *datafile* is a collection of records organized on a disk and identified by a *filename*.

A *record* is all the data that exists between (but *not* including) the record delimiters.

A *record delimiter* is a character (usually an ASCII carriage-return character) that separates one record from the next in a datafile.

An *end-of-file delimiter*, or EOF, is the data written to a file that signifies there are no more records in the file.

Finally, a record may be subdivided into data fields, so a record may also contain a *data-field delimiter*, which is usually an ASCII comma character.

Relative files earned their name because there is a definite relationship between the record number and its position within the file. This relationship can be defined because all records in a relative file *must* have the same length. This allows relative files to use a formula (characters per record × record number) that determines a record's location by counting the number of characters from the start of a file. Therefore, it is possible for the disk operating system to read the tenth record (or any other record) in the file without reading any prior records. It is also possible to change a record by writing the changed information into the record directly.

With all the apparent power of relative files, why would anyone want to use sequential files? Well, each type has its own advantages. Here are some.

Sequential files: maximum file utilization, no fixed record size; larger total file space per disk; more total records possible per disk.

Relative files: records can be found by number without reading other records; individual records can be updated; records can be subdivided into individual fields; individual data fields within records can be retrieved with special Basic statements.

It may not be immediately apparent what type of data would best be suited for each type of file, so a little explanation is in order. The type of data that would best be saved with a sequential file is historical data and data that requires only occasional retrieval.

A specific example of historical data might be a daily record of expenses and taxes paid, for income tax purposes. After it is correctly entered, it's unlikely that this information would change. The type of data that would best be saved in a relative file would be data that constantly requires updating or frequent reference; information about a stock portfolio, for example.

The Intelligent 1541 and 1571

Writing programs to store data on a 1541 or 1571 disk drive requires an understanding of some specific disk commands. Once you master them, however, you gain a very versatile DOS. This is because the 1541 or 1571 disk drive, and other Commodore-compatible drives are intelligent peripherals; they have their own microprocessors and both use a *serial* communication interface. Both have a primary address (usually eight, which is the device number), and may have several secondary addresses for establishing a command/error channel and data channels. The problem with using device address and secondary address is that commands must be sent and errors checked using somewhat complex Basic programming.

This article is meant to clarify and demystify the data storage techniques of the 1541 or 1571 disk drive and other Commodore-compatible disk drives.

Sequential File Experiments

Now that you have a better idea of the two general types of datafiles, let's investigate and experiment with the sequential datafile. For this, you should have an old unformatted disk. To format a blank disk, you type in the following Basic command.

```
OPEN 15,8,15,"N0:TESTDISK,##":CLOSE 15
```

To write records into a sequential file, the file must be opened and given a filename, a file number, a drive number and a data channel. Records must then be written into the file, and, finally, the file must be closed. The filename is used by the DOS to locate the file when you later want to read records from it. The file number is used by Basic so that, by sending record data to the proper data channel, it can write the records into the proper file. You should

choose a file number less than 128, because larger file numbers will place, after each record, an extra record limiter (an ASCII line feed), which might cause some problems, particularly with end-of-file detection.

To experiment with writing data to a sequential file, enter and save the following program, known as SEQWRITE1.

```
1 REM **LISTING 1—SEQWRITE1**
10 OPEN 15,8,15, "S0:TAXLOG" :CLOSE15:REM ERASES TAXLOG FILE
20 OPEN 2,8,2, "0:TAXLOG,S,W"
30 D$ = "H REX, MD":GOSUB 1000
40 D$ = "1/18/86":GOSUB 1000
50 D$ = "139.28":GOSUB 1000
60 CLOSE 2:END
1000 PRINT#2, D$:RETURN
```

Look at the Open statement in line 20; this opens the file. The first number (2) refers to the file number that is used later. The second number refers to the disk device (8). The third number refers to the data channel (2) that will be used to send the data to the file. For convenience, the data channel is usually chosen to be the same number as the file number, although the data channel may be any number from 2 to 14. Henceforth, any future reference to file number 2 will be allocated as such: Anything written to file 2 is sent on channel 2 to device 8.

The data that is enclosed in quotes behind the Open statement's file, device and channel number parameters is usually referred to as the filename; however, it contains much more information than just the filename. It is the actual information sent to the disk drive to set up the sequential file, and it's more properly referred to as a *file parameter string.*

The actual filename is TAXLOG, and the ",S,W" indicates to the DOS that you want to open a sequential file (S) and that you want to write to it (W).

The Print#2 statement on line 1000 writes a record into file 2, the sequential file opened in line 20. Writing records into a file is much like printing a string to the disk drive.

Remember the data-path allocation plan set up by the Open statement: Anything written to file 2 is sent on channel 2 to device 8. The 1541 or 1571 DOS knows that channel 2 will have incoming data for the TAXLOG file. Everything works according to a plan. It is also necessary to close the file. The Close 2 statement tells the

DOS to write an EOF delimiter in the TAXLOG file and to cancel the file 2 data-path allocation plan.

Run the program. The record data will be written into the file in the following sequence:

```
H REX, MD*1/18/86*139.28*
```

Note: "*" represents an ASCII carriage-return character.

To read records from a file, the file must first be opened with a filename, a file number, a disk drive number and a channel number. The records in the file are read, and then the file is closed. Type in the following program and save it as SEQREAD1.

```
1 REM **LISTING 2—SEQREAD1**
10 OPEN 2,8,2, "0:TAXLOG,S,R"
20 GOSUB 1000:NA$ = D$:GOSUB1000:DA$ = D$
30 GOSUB 1000:A = VAL(D$)
40 CLOSE 2
60 PRINT NA$:PRINT DA$:PRINT A:END
1000 INPUT#2, D$:RETURN
```

The Open statement sets up a data-path allocation plan for file 2 to be allocated to device 8 and channel 2. When you open a sequential file for reading, the filename parameter string can contain either the filename alone or the filename and "0:" drive identifier. The absence of the ",S,R" suffix to the filename parameter string tells the DOS to assume a sequential file and a read access. The DOS locates the file by filename in the directory and prepares to provide record data on channel 2.

The Input# Technique

To read the records in the file, the Input# statement is used. The Input# statement is similar to the Input statement, except that the Input# statement expects data from a previously opened file, rather than from the keyboard. Each record is read and assigned to a variable, with a conversion to a numeric variable when necessary. Finally, the file is closed with a Close statement and variables are printed. Run the program. You should see the following:

```
H REX
1/18/86
139.28
```

Note that the , MD is missing on the above. The record written

into the file with the SEQWRITE1 program was "H REX, MD", not "H REX". Actually, the entire record was read in, but the Input# statement recognized a comma in the input data and truncated the data transfer to the variable. This quirk of the Input# statement, while undesirable in this example, can be put to some use, because it enables you to input separate data fields in a single record.

When you use the Input# statement to read a record in a disk file, the DOS transfers the next record from the disk into the computer's Basic input buffer. The DOS continues to transfer characters from the record into the Basic input buffer until it reads a record terminator. The record read into the Basic input buffer corresponds exactly to what was written to the disk with the single Print# statement that wrote the record. After the DOS sees the record terminator (a carriage return), Basic then assigns the characters in the input buffer to a variable until it sees either the record terminator or a field terminator (a comma).

If the Input# statement has a list of variables separated by commas, and if there are commas in the record, then the next variable in the list will be assigned the characters in the next field of the record. To make use of data fields in records, there are some pitfalls you must avoid.

To write a record with data fields, you must place commas into the record between the fields of data. This can be done by setting a string variable equal to a comma (CM$ = ",") and writing the fields of data into the record using a single Print# statement as follows:

```
PRINT#2, F1$;CM$;F2$;CM$;F3$
```

Don't use a statement like "PRINT#2, F1$,F2$,F3$", because the data fields are not separated by commas. Also, you must be careful that the length of data in all the fields of the record (including the commas that delimit the fields) does not exceed 80 characters; if it does, the C-64's input buffer will overflow when you use the Input# statement to read the record. (The C-128's limit is 254 characters.)

To read a record with data fields, you must use only a single Input# statement per record, and the statement must have a variable list to indicate to Basic which variable will be used to store which data field. If the record has three data fields and you use the following statement, the data will be properly read from the disk.

```
INPUT#2, F1$,F2$,F3$
```

On the other hand, if you try to input the fields of data with

separate Input# statements, for example

```
INPUT#2,F1$
INPUT#2,F2$
```

the data read will not be correct because each new Input# statement reads a new record from the disk, regardless of whether or not all of the data fields of the previous record have been assigned to variables by Basic.

There are some characters that should not be used with the Input# statement. Commas, colons and null characters will cause problems because they separate fields of data.

Before assigning the data to a variable, the Input# statement removes all leading spaces; therefore, if you want to include spaces before your data, use commas.

The length of records cannot exceed 80 characters (or, in the case of the C-128, 254 characters), since the C-64 has only an 80-character input buffer. A String Too Long error will result when the Basic input buffer is overfilled by the DOS.

The leading space problem, the comma and colon restrictions, and the 80-character record-length restriction are somewhat limiting, particularly when records contain text, so special techniques are available to overcome these restrictions.

GET# IT TOGETHER

To include commas or colons in records, use the Get# command in place of the Input# command. The Get# technique is used to input data from records that are more than 80 characters in length. To experiment with this technique, load the SEQREAD1 program and make the changes and additions that are indicated by the underlined text in the following program, which you should save as SEQREAD2.

```
1 REM **LISTING 3—SEQREAD2**
10 OPEN 2,8,2, "0:TAXLOG,S,R"
20 GOSUB 1000:NA$ = D$:GOSUB 1000: DA$ = D$
30 GOSUB 1000:A = VAL(D$)
40 CLOSE 2
60 PRINT NA$:PRINT DA$:PRINT A:END
1000 D$ = ""
1010 GET#2, C$:IF ASC(C$) = 13 THEN RETURN
1020 D$ = D$ + C$:GOTO 1010
```

Run this program. The Get# technique eliminates the character restrictions. Normally, it would also eliminate the record-length limitations of the Input# technique. However, unlike the Input# command, which utilizes the computer's operating system, Get# uses Basic, which is substantially slower.

The Quoted-String Technique

The Input# technique can be used to input commas and colons in record data if the records are written into the file as a quoted string. This technique works only when the record length is less than 80 characters. In this technique, the quote character (ASCII value 34 decimal) is written as the first character of the record.

The Input# technique is then used to read the records. The quote character at the start of the record forces the Input# statement, which assigns all of the data read from the record, including commas and colons (but not the initial quote character), into a *string* variable. Load the SEQWRITE1 program and make the following changes and additions indicated by the underlined text.

```
1 REM **LISTING 4—SEQWRITE2**
10 OPEN 15,8,15,"S0:TAXLOG" :CLOSE15:REM ERASES TAXLOG FILE
20 OPEN 2,8,2,"0:TAXLOG,S,W"
30 D$="H REX, MD":GOSUB 1000
40 D$="1/18/86":GOSUB 1000
50 D$="139.28":GOSUB 1000
60 CLOSE 2:END
1000 PRINT#2, CHR$(34);D$:RETURN
```

Save this program as SEQWRITE2 and run it. The earlier version of TAXLOG is erased and new data is now stored in the file as follows:

```
"H REX, MD*"1/18/86*"139.28*
```

Note: "*" represents an ASCII carriage-return character.

Since each record is preceded by a quotation mark, the records must be input as *string* variables, since Basic cannot input a numeric variable, even if it is preceded by a quotation mark. If the data contains numeric information, the string can be converted to a number using the VAL function. Load the SEQREAD1 progam and run it.

Note that the Input# statement now inputs a comma because of the initial quotation mark written to the record before the actual

data. While the SEQWRITE2 program writes the data in a manner that permits the input of commas, colons and leading spaces, it will not permit the input of a quotation mark.

The SEQWRITE2 program writes one more character to each record than the SEQWRITE1 program, so the maximum number of records that will fit on a disk using the SEQWRITE2 program will be less than with the SEQWRITE1 program.

Detecting the End of the File

In practice, a sequential file will contain an unknown number of records. Therefore, you will need to know when the last record has been read from the file.

Basic provides a reserved variable, ST, to monitor the status of disk drive input/output operations. Using the Input# and Get# technique, the ST variable detects when the last record in the file is read. The next listing, SEQREAD3, tests the status variable to see if it checks for the last record.

```
1 REM **LISTING 5—SEQREAD3**
10 OPEN 2,8,2, "0:TAXLOG,S,R"
20 GOSUB 1000:PRINT "STATUS = "S;D$
30 IF S AND 64 = 0 THEN 20
40 CLOSE 2:END
1000 INPUT#2, D$:S = ST:RETURN
```

Enter the program and save it as SEQREAD3. Run the program and note that it stops automatically after reading the three records in the "TAXLOG" file.

Adding Records

It's easy to add more records to a sequential file. Load the SEQWRITE2 program and make the changes indicated by the underlining in the listing below. Save the program as SEQWRITE3.

```
1 REM **LISTING 6—SEQWRITE3**
10 REM
20 OPEN 2,8,2, "0:TAXLOG,S,A"
30 D$ = "H REX, MD":GOSUB 1000
40 D$ = "1/18/86":GOSUB 1000
50 D$ = "139.28":GOSUB 1000
60 CLOSE 2:END
1000 PRINT#2, CHR$(34);D$:RETURN
```

The ",S,W" portion of the filename parameter string is replaced with

",S,A", which tells the DOS that you want to open an existing sequential file (S) and that you want to *append* (A) data to the file.

Run the SEQWRITE3 program. Every time you run it, it adds three more records to the TAXLOG sequential file. Load and run the SEQREAD3 program to verify the proper data content of the TAXLOG file.

The Append mode makes it easy to feed information into files in, say, monthly batches. Of course, the program you use would have to open and create the sequential file the first time you run the program and to use the ",S,A" parameters thereafter.

But how will the program know when to create and when to append? The program could first try to open the file in the Append mode, then, if it found that the file doesn't exist, it could switch to the Create mode and create the file. To do this, however, it is necessary for the program to be able to read DOS errors on the command/error channel.

Trapping Disk-Related Errors

At one time or another, while using the 1541 disk drive, you've probably seen its red light flash (it's a flashing green light on the 1571). This flashing indicates that some type of error has occurred and that the disk drive is ready to tell you what type of error.

Disk errors are detected by reading the command/error channel, which is channel 15. To read this channel, open a file to it and input the error data. The error data consists of four separate types of information: the error number, a description of the error and two more miscellaneous numbers. If you've made a major error, those last two numbers will be track and sector numbers that just might permit someone to restore the data on the disk.

The following listing, the SEQREAD4 program, contains a general-purpose, error-trapping routine that displays disk errors when they occur. Load the SEQREAD3 program and add or change the underlined text; then save the program as SEQREAD4.

```
1 REM **LISTING 7—SEQREAD4**
5 OPEN 15,8,15:REM ERROR CHANNEL WILL BE FILE 15
10 OPEN 2,8,2, "0:TAXLOG,S,R" :GOSUB2000
20 GOSUB 1000: PRINT "STATUS = " ;S;D$
30 IF S AND 64 = 0 THEN 20
40 CLOSE 2:CLOSE 15:END
1000 INPUT #2, D$:S = ST:GOSUB 2000: RETURN
```

```
2000 INPUT #20, E,ER$,T,S
2020 IF E<20, THEN RETURN
2030 PRINT E;ER$;T;S:STOP
```

Line 5 opens a file that will be used for input on the command/error channel, channel 15. Note that every disk-access statement has a call to subroutine 2000, which inputs and then checks the disk-error parameters. The error parameters will be printed out, and the program will stop if there is an important error.

Run the SEQWRITE3 program and see how many errors you can generate. Table 1 shows a list of the common user errors and the associated disk error.

Wrapping It Up

On the ReRUN disk, you'll find a program entitled SEQ/FILE/HANDLER, which contains a fully commented program that's useful for designing your own data-handling programs. The subroutines from lines 390 to 990 are used to perform and check disk input/output operations, while the main body of the program demonstrates these functions.

The file-handling routine, which checks to see if a disk is in the drive, uses the Initialize command. When the disk is initialized, the block allocation map (BAM) is loaded into the drive's DOS. The Initialization command checks to see if the disk is seated properly and if it's formatted and readable.

The file-handling routine simply tries to open the file for reading and then checks for errors. When a file is created, appended or

Table 1. Common user errors and related disk errors.

Common user error	Disk drive error number and message
*No disk in disk drive	74 DRIVE NOT READY
*Unformatted disk in disk drive	74 DRIVE NOT READY
*Disk in drive upside down	74 DRIVE NOT READY
*Disk drive door open	74 DRIVE NOT READY
Wrong disk in disk drive	62 FILE NOT FOUND
Disk write-protected	26 WRITE PROTECT ON

*Strangely, these conditions may also generate a 21 READ ERROR on track 18, sector 0.

scratched (erased), the disk must not be write-protected. Each of these functions must check for error 26, "Write Protect On," and proceed accordingly. Finally, if the file parameter string is a string expression instead of a quoted string, then it must be enclosed by parentheses to force Basic to send the entire expression to the disk drive.

Use this article as a stepping-stone to explore further sequential files and to create your own programs. The more you work with sequential files, the easier they'll be for you to handle.

RUN it right: **C-64**

The Remarkable Disk Directory

By James Hoffer

ANNOTATED DIRECTORY PRINTER is a program that will dramatically increase the usefulness of your disk directories. With it, you can add comments, references, program descriptions, special loading instructions, SYS commands, or almost anything else you wish, to the directory. This information appears in the directory display or printout right along with the usual directory information of file length, filename and filetype.

The sequential file that Annotated Directory Printer (A.D.P.) creates can be saved on the same disk you're annotating, and its name will be the same as the name of the disk. You can add to or modify the file as needed.

A.D.P.'s main menu appears in Table 1. In using the program's various functions, be sure to follow the instructions carefully.

Choose option 4 to create a new file. If you have no comment about a particular directory entry, the program will automatically leave the comment area for that item blank when printing out the directory.

When you choose option 3 to update a file, the merge screen will

Table 1. The A.D.P. main menu.

1. Load disk directory.
2. Load annotated directory.
3. Merge directories.
4. Display entries, add notes.
5. Print annotated directory.
6. End program.

remind you to superimpose the normal directory over the sequential file. Then, using option 4, you can add comments to each new item in the directory and change other items at will. A merge, however, will only be successful if the order of the items in the directory has not been altered—in other words, if you have only added new programs to the end of the list.

To print out the annotated directory, follow the prompts in option 5. You can provide alternative typestyles for the printout by modifying A.D.P. to include appropriate codes for your printer. An alternative typestyle would be helpful, for instance, for preparing a small archive printout to be kept right in the disk envelope.

I used a few tricks in writing A.D.P. that might interest you if you're a programmer. Note the screen and color Pokes that produce the green arrows next to the main menu and the asterisk used as an end-of-file marker. Also, I dimensioned all the string variables to 144, the maximum number of directory entries.

RUN it right: **C-64 or C-128; printer**

Mom's Kitchen Aid

By Lon Olson

AS ANY HOMEMAKER KNOWS, the kitchen is a hectic place, especially at mealtime. Mom's Kitchen Aid won't exactly slow the pace, but it will help get things in order. Two programs in one, it

prints out blank forms for both menus and shopping lists. The one-page menu form has room for a week's meals; the shopping list form is filled with four blanks that, when cut apart, are just the right size to carry to the store.

You may want to post the forms on the kitchen bulletin board or the refrigerator door, so family members can see what meals are scheduled and make additions to the grocery list.

Mom's Kitchen Aid works in 64 or 128 mode and supports virtually any printer. Naturally, take care to type in the lines exactly as shown, and save a copy of the program to disk before running it.

After running the program, you're prompted for the name you want printed on each form. If you want Mom, just press the return key. Next, enter 1 or 2 to designate the form to be printed, first making sure that your printer is turned on and the paper is aligned with the top of the printhead.

Lines 60–80 of the program contain three printer codes that work with most Commodore and third-party printers. If you have a 1526 or MPS-802 that uses the OPEN 6,4,6 : PRINT#6,CHR$(XX) format for line spacing, remove the first REM in line 330 to bring your lines closer together. Everything else remains the same.

RUN it right: **C-64**

Medium Resolution Graphics and Character Enlarger

By Roger Moore

MEDIUM RESOLUTION GRAPHICS AND CHARACTER ENLARGER is a handy, interrupt-driven subroutine that turns the C-64's text screen into an 80 × 50 matrix on which you can plot (or erase) points to create your own graphics, and you can enlarge letters and keyboard graphics up to 16 times normal size. The utility also displays

text and graphics together. Just run "MED-RES GRAPHICS" to activate it.

To plot or erase a point anywhere on the grid, enter:

```
SYS49152,X,Y,DF,COLOR
```

where SYS49152 executes the program, X,Y are the x and y coordinates of the grid, DF is the draw flag that plots (use 1) or erases (use 0) a point. COLOR can be any value from 0–15. For instance, SYS49152,32,12,1,7 puts a yellow point at coordinates 32,12.

Plotted points are in the foreground color. Because of color RAM limitations, any points occupying the same cursor block will be of one color; otherwise you can plot all 16 colors on the screen.

Coordinates originate in the upper-left corner of the screen (0,0). The upper-right corner coordinates are (79,0), the lower-left corner is (0,49), and the lower-right corner is (79,49).

The Character Expander routine makes letters or keyboard graphics 16 times normal size. You execute the routine with:

```
SYS49155,X,Y,COLOR,CHARCODE,DF
```

Again, X,Y are the screen locations, with X ranging from 0–39 and Y ranging from 0–49. COLOR and DF are used the same way as in the plot command. CHARCODE is the screen Poke value for the character, *not* the ASCII value. Run "MED-RES DEMO" for examples of using the routine.

RUN it right: **C-128 mode; disk drive**

1581 Directory

By Robert S. Bloemer

COMMODORE'S 1581 DISK DRIVE is superb in providing the speed and capacity you need to fully utilize the powers of the C-128. On a single, 3½-inch disk you can store not only a variety of programs, but also the data files associated with them, thus making disk-swapping virtually obsolete. To better manage the disk's large capacity (808K), the disk operating system provides a new feature: the partitioning, or dividing, of the disk into user-defined areas so that the various

programs can be stored separately. You can even save CP/M programs along with regular Commodore programs on a single disk.

The partitioning feature is great, but, unfortunately, Commodore did not provide a DOS Shell for the 1581, and the familiar 1571 DOS Shell will not work with it. The utility disk that comes with the 1581 does have a program that creates partitions and another that copies files into them, but once the partitions are in place, there's no easy way to access the sub-directories on the divided disk.

Coming across the machine language directory program by Jon R. Curtis in *RUN*'s Magic column of November 1987 (trick $43C, p. 12), I immediately recognized it would be a splendid core for a full-blown directory program that would meet my requirements of ease, flexibility and speed.

The result is 1581 Directory. Written in Basic 7.0 for C-128 mode only, it can be used with 40- or 80-column screens and with any combination of drives, from the 1541 to the 1581. To make it most effective, I save 1581 Directory as the first program on *all* disks and as the first program in each *partition* of my 1581 disks.

In addition, I designate it as the auto-boot program on the disk. (Be sure *not* to partition track 1 if you want auto-booting.) This makes 1581 Directory present when you power up the computer and available after running another program by simply pressing the shifted run-stop key. After you select and use another program, booting 1581 Directory returns it to the last-used drive number, and, if the drive is a 1581, to the sub-directory you last selected.

Uses and Menu Options

The 1581 Directory program sorts a directory into the different types of files, from the new 1581 "CBM" file, which titles the various sub-directories, to program, sequential and relative files—all presented so as to maximize the number of files shown on a single screen (72 on the 80-column screen). When you're using the 1581 drive, the CBM files are presented first, so that you can access the subdivisions quickly. With other drives, it lists the program files first to allow easy selection and running thereof. To change drive directories, you simply press the C key, and the program defaults to toggling between drives 8 and 9.

At the bottom of the screen is a menu offering the options of selecting a program to run, changing file types, changing drive numbers or exiting the directory. Pressing the D key presents you with a menu of disk operating commands. The options available

here are renaming files, deleting files and formatting disks, as well as a boot option to run machine language programs. I've also included a simple sequential file reader for identifying a specific file among a number that are similarly named.

There are three additional functions available that are not listed on any menu because of screen clutter. The up-arrow key (↑) lets you repeat large file listings that use more than one screen; the N key tells you program size and blocks remaining; the X key toggles between 40- and 80-column modes while at the same time preserving the directory screen. These three functions are available from either menu.

Other Features

One of the most useful features of 1581 Directory is that when you're changing drives, the original drive is left in the partition last selected. This allows you to load a program from one drive and save it in the partition of the other drive. When you rerun the directory program, you are returned to the desired partition.

Numerous safeguards are incorporated to prevent the program from crashing when you're turning drives on and off or inserting unformatted disks. It's not foolproof, but most unusual activities are covered. Since the machine language portion of the program resides in the RS-232 buffer, some ML programs, most notably terminal programs, can cause trouble when you're rerunning the directory program. At the worst, a cold reset may be necessary. Sometimes a Poke 3075,0 and running can restore normalcy.

One other note: The initial running of the program checks for the presence of drives 8 and 9, notes their model numbers (1581, 1571, etc.) and data-reads the ML program to $0C00. These steps take about four to six seconds. Subsequent loads and runs of the program just check the presence of the ML program and the drive information poked into memory. If it finds them intact, it bypasses the rest of the checking process and boots up much faster. In either case, the program is faster than the 1571 DOS Shell.

Using this program, I have managed to store, on five 3½-inch disks, all my important programs that were formerly contained on some 90 5¼-inch disks, although most of these were, admittedly, less than full. What's more, because of partitioning and the 1581 Directory program, all is organized in a format that makes it easy to find and select any desired program. It has truly made my computing sessions a delight!

RUN it right: **C-64; C-128 (in C-64 mode)**

Keys to Convenience

By Keven J. Stone

UNLIKE MOST COMPUTERS ON THE MARKET, the C-64 lets you redefine a key or reconfigure the entire keyboard, providing great flexibility for various programming needs.

I have used redefined keyboards with various commercial software packages and find the added flexibility to be a great aid. For example, my spreadsheet program is much easier to use with a numeric keypad, but, unfortunately, the C-64 is not equipped with one.

To get around this problem, I simply substituted the graphics symbols for a numeric keypad. When needed, a simple press of the shift-lock key calls the pad into operation; otherwise, the keyboard functions normally. The procedure is:

1) LOAD "KEYPAD.KB",8,1
2) SYS 49152
3) NEW
4) LOAD "SPREADSHEET",8
5) RUN

The Keyboard Editor program accesses the keyboard and lets you manually generate reconfigured keyboard files, which you can later load and use in various configurations. Since Keyboard Editor accomplishes the reconfiguration process via a hardware feature of the C-64, almost all programs will work normally with a redefined keyboard in place.

Using the Program

After loading and running Keyboard Editor, a menu with the following options will be displayed:

F1—RECONFIGURE KYBD
F3—TEST KYBD
F5—SAVE KYBD
F6—LOAD KYBD
F7—QUIT

F1—RECONFIGURE KYBD allows you to reconfigure individual keys on the C-64 keyboard. To change a key, answer the two prompts that appear. The first will ask for the key that is to be changed (press it), and the second will ask for the key you wish to substitute (press it). When the keyboard has been changed to your satisfaction, press function key f7 to return to the main menu.

F3—TEST KYBD enters the Typewriter mode to allow testing of the keyboard file currently in memory. Keypresses will be echoed to the screen. Any corrections to the keyboard file can be made by exiting the Typewriter mode with f7 and again choosing menu item 1.

F5—SAVE KYBD. With a valid, user-supplied name, this saves the current keyboard file in memory to the disk drive. The file will be saved in such a manner that it can be loaded and used without running Keyboard Editor. To abort the save, press the return key when prompted for a filename. At the end of a successful save, you'll be returned to the main menu. To easily identify keyboard files on disk, place ".KB" as the last three letters of the filename.

F6—LOAD KYBD loads a previously saved keyboard file back into memory, where it can be tested or further edited. To abort the load, press the return key when prompted for a filename. If the load is successful, you'll be returned to the main menu.

F7—QUIT returns you to the Basic environment, leaving Keyboard Editor resident in memory.

Because Keyboard Editor saves keyboard files in such a way that they can be loaded directly, you can use a keyboard file in several different ways: in Direct mode; in a Basic program; or with commercial software.

To use a keyboard file in the Direct mode, perform the following steps:

a) LOAD "filename.Kb",8,1
b) SYS 49152
c) NEW

The keyboard file will now be in place of the normal C-64 keyboard. With the addition of one more step, the above procedure also holds true when you use the keyboard file with commercial software. Simply load and run the commercial software normally.

To use a keyboard file from within a Basic program, add the following lines to the beginning:

```
10 IF A=0 THEN A=1:LOAD "filename.Kb",8,1
20 SYS 49152
```

This will load the keyboard file from the disk drive into memory and perform the reconfiguration. Placing SYS 49152 commands at various points in the program allows you to toggle between the regular C-64 keyboard and the user keyboard file. The only restriction is that the Basic program must not use the free RAM memory at $C000 or the zero-page locations $FD and $FE, which are used as flags for the machine language portion of Keyboard Editor. If you're not going to toggle back to the regular keyboard, then this restriction does not apply. ■

TECHNICAL EDITOR
TIM WALSH

MANAGING EDITOR
SWAIN PRATT

COPY EDITOR
PEG LEPAGE

PROOFREADER
HAROLD BJORNSEN

DESIGN AND LAYOUT
ANNE DILLON

TYPESETTING
DEBRA DAVIES
KEN SUTCLIFFE

RERUN MANAGER
PAUL FINCH

CUSTOMER SERVICE REPRESENTATIVE
DEBBIE BOURGAULT

18 Programs Included on this Disk

Alphabet Cadet ► Color Cornucopia ► Bill Minder
Travel Tally ► Loan Analysis ► Video Poker Game
Adventure Game ► Disk File Helper ► Kitchen Aid

From the July RUN:

- ► Alphabet Cadet
- ► The Sixteen-Color Print Machine
- ► Color Cornucopia
- ► Bill Minder
- ► Travel Tally
- ► Evel's Revenge
- ► Loan Analysis
- ► Disk Restorer

From the August RUN:

- ► Islands
- ► Net Worth Calculator
- ► Video Poker
- ► Disk File Helper
- ► Sequential File Design
- ► The Remarkable Disk Directory
- ► Mom's Kitchen Helper
- ► Medium Resolution Graphics

Plus: Extra Bonus Programs!

- ► 1581 Directory
- ► Keys to Convenience

If any manufacturing defect becomes apparent, the defective disk will be replaced free of charge if returned by prepaid mail within 30 days of purchase. Send it, with a letter specifying the defect, to:

ReRUN • 80 Elm Street • Peterborough, NH 03458

Replacements will not be made if the disk has been altered, repaired or misused through negligence, or if it shows signs of excessive wear or is damaged by equipment.

The programs in ReRUN are taken directly from listings prepared to accompany articles in *RUN* magazine. They will not run under all system configurations. Use the RUN It Right information included with each article as your guide.

The entire contents are copyrighted 1988 by IDG Communications/Peterborough. Unauthorized duplication is a violation of applicable laws.

©Copyright 1988 IDG Communications/Peterborough